

leetcode Count of Smaller Numbers After Self

📅 2015年12月6日 (<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/>) 👤 hrwhisper (<https://www.hrwhisper.me/author/hrsay/>) 💬 4 Comments (<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/#comments>) 👁 5,449 views

leetcode Count of Smaller Numbers After Self

You are given an integer array *nums* and you have to return a new *counts* array. The *counts* array has the property where `counts[i]` is the number of smaller elements to the right of `nums[i]`.

Example:

Given `nums = [5, 2, 6, 1]`

To the right of 5 there are 2 smaller elements (2 and 1).

To the right of 2 there is only 1 smaller element (1).

To the right of 6 there is 1 smaller element (1).

To the right of 1 there is 0 smaller element.

Return the array `[2, 1, 1, 0]`.

题目地址 leetcode Count of Smaller Numbers After Self

(<https://leetcode.com/problems/count-of-smaller-numbers-after-self/>)

题意:

给定nums数组, 求数组中每个元素i的右边比其小的数

思路:

简单的说就是求逆序数。

1. 使用逆序数有经典的解法为合并排序。

2. 用Fenwick树 关于Fenwick 树介绍 Binary indexed tree (Fenwick tree)

(<https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/>)

- 简单说就是看当前数在nums中排第几，然后对小于它的数求个数和
- 具体的做法是先离散化，确定每个数在nums中排到第几（去重和排序）
- 然后从右向左扫描，每次统计比其小于1的个数（就是求和），然后把当前的数加入Fenwick中。

merge_sort

C++

```
1 struct Node {
2     int val;
3     int index;
4     int cnt;
5     Node(int val, int index) : val(val), index(index), cnt(0) {}
6     bool operator <= (const Node &node2) const {
7         return val <= node2.val;
8     }
9 };
10
11 class Solution {
12 public:
13     void combine(vector<Node> &nums, int Lpos, int Lend, int Rend, vector<No
14         int Rpos = Lend + 1;
15         int Tpos = Lpos;
16         int n = Rend - Lpos + 1;
17         int t = Rpos;
18         while (Lpos <= Lend && Rpos <= Rend) {
19             if (nums[Lpos] <= nums[Rpos]) {
20                 temp[Tpos] = nums[Lpos];
21                 temp[Tpos].cnt += Rpos - t ;
22                 Tpos++; Lpos++;
23             }
24             else {
```

```

25         temp[Tpos++] = nums[Rpos++];
26     }
27 }
28
29 while (Lpos <= Lend) {
30     temp[Tpos] = nums[Lpos];
31     temp[Tpos].cnt += Rpos - t;
32     Tpos++; Lpos++;
33 }
34
35 while (Rpos <= Rend)
36     temp[Tpos++] = nums[Rpos++];
37
38 for (int i = 0; i < n; i++, Rend--)
39     nums[Rend] = temp[Rend];
40 }
41
42 void merge_sort(vector<Node> & nums, int L, int R, vector<Node> &temp) {
43     if (L < R) {
44         int m = (L + R) >> 1;
45         merge_sort(nums, L, m, temp);
46         merge_sort(nums, m + 1, R, temp);
47         combine(nums, L, m, R, temp);
48     }
49 }
50
51 vector<int> countSmaller(vector<int>& nums) {
52     vector<Node> mynums;
53     vector<Node> temp(nums.size(), Node(0, 0));
54     for (int i = 0; i < nums.size(); i++)
55         mynums.push_back(Node(nums[i], i));
56
57     vector<int> ans(nums.size(), 0);
58     merge_sort(mynums, 0, nums.size() - 1, temp);
59
60     for (int i = 0; i < nums.size(); i++)
61         ans[mynums[i].index] = mynums[i].cnt;
62
63     return ans;
64 }

```

```
65 };
```

Binary indexed tree (Fenwick tree)

C++

```
1  class FenwickTree {
2      vector<int> sum_array;
3      int n;
4      inline int lowbit(int x) {
5          return x & -x;
6      }
7
8  public:
9      FenwickTree(int n) :n(n), sum_array(n + 1, 0) {}
10
11     void add(int x, int val) {
12         while (x <= n) {
13             sum_array[x] += val;
14             x += lowbit(x);
15         }
16     }
17
18     int sum(int x) {
19         int res = 0;
20         while (x > 0) {
21             res += sum_array[x];
22             x -= lowbit(x);
23         }
24         return res;
25     }
26 };
27
28 class Solution {
29 public:
```

```

30     vector<int> countSmaller(vector<int>& nums) {
31         vector<int> temp_num = nums;
32         sort(temp_num.begin(), temp_num.end());
33         unordered_map<int,int> dic;
34         for (int i = 0; i < temp_num.size(); i++)
35             dic[temp_num[i]] = i + 1;
36
37         FenwickTree tree(nums.size());
38         vector<int> ans(nums.size(),0);
39         for (int i = nums.size() - 1; i >= 0; i--) {
40             ans[i] = tree.sum(dic[nums[i]] - 1);
41             tree.add(dic[nums[i]],1);
42         }
43         return ans;
44     }
45 };

```

Python

```

1  class FenwickTree(object):
2      def __init__(self, n):
3          self.sum_array = [0] * (n + 1)
4          self.n = n
5
6      def lowbit(self, x):
7          return x & -x
8
9      def add(self, x, val):
10         while x <= self.n:
11             self.sum_array[x] += val
12             x += self.lowbit(x)
13
14     def sum(self, x):
15         res = 0
16         while x > 0:
17             res += self.sum_array[x]
18             x -= self.lowbit(x)
19         return res

```

```

20
21
22 class Solution(object):
23     def countSmaller(self, nums):
24         """
25         :type nums: List[int]
26         :rtype: List[int]
27         """
28         dic = {}
29         for i, num in enumerate(sorted(list(set(nums)))):
30             dic[num] = i + 1
31         tree = FenwickTree(len(nums))
32         ans = [0] * len(nums)
33         for i in xrange(len(nums) - 1, -1, -1):
34             ans[i] = tree.sum(dic[nums[i]] - 1)
35             tree.add(dic[nums[i]], 1)
36         return ans

```

本博客若无特殊说明则由 hrwhisper (<https://www.hrwhisper.me>) 原创发布
 转载请点名出处：细语呢喃 (<https://www.hrwhisper.me>) > leetcode Count
 of Smaller Numbers After Self ([https://www.hrwhisper.me/leetcode-count-
 of-smaller-numbers-after-self/](https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/))

本文地址：[https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-
 after-self/](https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/) ([https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-
 after-self/](https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-

 after-self/))

分享到： 新浪微博 微信 QQ空间 人人 腾讯微博 豆瓣 印象笔记

📁 Leetcode (<https://www.hrwhisper.me/category/code/leetcode/>) 🔖 c++

(<https://www.hrwhisper.me/tag/c/>), leetcode (<https://www.hrwhisper.me/tag/leetcode/>), python

(<https://www.hrwhisper.me/tag/python/>), 算法 (<https://www.hrwhisper.me/tag/algorithm/>). 🔗 permalink

(<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/>).

◀ windows 10 远程桌面提示凭证无法工作解决办法 (<https://www.hrwhisper.me/windows-10-remote-desktop-credential-not-work-solution/>)

Binary indexed tree (Fenwick tree) ▶ (<https://www.hrwhisper.me/binary-indexed-tree-fenwick-tree/>)

4 thoughts on “leetcode Count of Smaller Numbers After Self”



Xiaoye says:

2016年5月11日 at pm3:11 (<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/#comment-998>)

你好博主，能不能对

```
for i in xrange(len(nums) - 1, -1, -1):  
    ans[i] = tree.sum(dic[nums[i]] - 1)  
    tree.add(dic[nums[i]], 1)
```

这段循环做个注释呢？
我看不懂里面的逻辑.....

<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/?replytocom=998#respond>



hrwhisper (<https://www.hrwhisper.me>) says:

2016年5月11日 at pm7:28 (<https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/#comment-1002>)

从右向左扫描，每次统计比其小于1的个数（就是求和），然后把当前的数加入Fenwick中（当前数在序列中排的位置（因为前面离散化了）个数加1）。

[hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/?replytocom=1002#respond](https://www.hrwhisper.me/leetcode-count-of-smaller-numbers-after-self/?replytocom=1002#respond)

Pingback: leetcode Count of Range Sum - 细语呢喃 (<https://www.hrwhisper.me/leetcode-count-of-range-sum/>)

Pingback: leetcode — Count of Smaller Numbers After Self — 经典求逆序数-IT大道 (<http://www.itdadao.com/article/74301/>)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

 (<http://weibo.com/murmured>)

 (<https://twitter.com/hrwhisper>)

 (https://instagram.com/hr_say/)

 (<https://github.com/hrwhisper>)

 (<https://www.hrwhisper.me/feed>)

Csdn (<http://blog.csdn.net/murmured>)

博客园 (<http://www.cnblogs.com/murmured/>)

Lofter (<http://hrsay.lofter.com/>)

知乎 (<http://www.zhihu.com/people/hrwhisper>)

豆瓣 (<http://www.douban.com/people/hrwhisper/>)

努力的人本身就有奇迹 | 快乐是我们共同的信仰

by hrwhisper.me